# STRNCPY

Make sure the buffer and bounds are the proper size to hold the source string plus a NULL character.

Sean Barnum, Cigital, Inc. [vita[1]]

2007-04-23

# Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 7939 bytes

<table>
<tr><td><b>Attack Category</b></td><td>• Malicious Input<br>• Denial of Service</td></tr>
<tr><td><b>Vulnerability Category</b></td><td>• Buffer Overflow<br>• No Null Termination</td></tr>
<tr><td><b>Software Context</b></td><td>• String Management</td></tr>
<tr><td><b>Location</b></td><td></td></tr>
<tr><td><b>Description</b></td><td>strncpy() and related functions copy a specific number of characters from one buffer to another. While the presence of the bound makes it safer than the similar strcpy function, it can still cause a buffer overflow.<br>The strncpy() functions are preferable to strcpy() because they accept boundaries for buffers that can be checked against. However, they are still vulnerable to certain attacks if used improperly:<br>1. passing of NULL for src or dest causes exception<br>2. 'count' size parameter is often incorrectly passed in<br>3. not guaranteed to have null terminated string upon exit<br>Make sure the buffer and bounds are the proper size to hold the source string plus a NULL character.</td></tr>
<tr><td><b>APIs</b></td><td><table>
<tr><th>Function Name</th><th>Comments</th></tr>
<tr><td>_mbsncpy</td><td></td></tr>
<tr><td>_tcsncpy</td><td></td></tr>
<tr><td>lstrcpyn</td><td>Windows</td></tr>
<tr><td>lstrcpynA</td><td></td></tr>
<tr><td>lstrcpynW</td><td></td></tr>
<tr><td>StrCpyN</td><td>"StrCpy" routines are from shell, Shlwapi.dll</td></tr>
<tr><td>StrCpyNA</td><td>"StrCpy" routines are from shell, Shlwapi.dll</td></tr>
</table></td></tr>
</table>

[1] http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

<table>
<tr><td></td><td>
<table>
<tr><td>StrCpyNW</td><td>"StrCpy" routines are from shell, Shlwapi.dll</td></tr>
<tr><td>StrCpyNW</td><td></td></tr>
<tr><td>StrNCpy</td><td>macro that calls the StrCpyN function</td></tr>
<tr><td>strncpy</td><td>make sure null terminated</td></tr>
<tr><td>strncpy</td><td></td></tr>
<tr><td>ualstrcpyn</td><td>unaligned Unicode characters on MIPS, PPC, Alpha</td></tr>
<tr><td>ualstrcpynA</td><td>unaligned Unicode characters on MIPS, PPC, Alpha</td></tr>
<tr><td>ualstrcpynW</td><td>unaligned Unicode characters on MIPS, PPC, Alpha</td></tr>
<tr><td>wcsncpy</td><td></td></tr>
</table>
</td></tr>
<tr><td><b>Method of Attack</b></td><td>An attacker can manipulate the input strings to cause access violations and possibly take control of the program. Passing NULL as src or dest can easily cause the program to terminate, thereby enabling a DoS attack. In some cases, passing in exactly the right size string can cause the resultant dest string to not be null terminated. This can potentially lead the further uses of the dest string to overflow into adjoining memory and cause buffer overflows. The most common problem, however, is improperly passing in the 'count' or 'length' parameter for strncpy, thus causing other buffer overflow problems. This is especially common when using wide double byte (Unicode) characters. Buffer overflows commonly occur with this function when the maximum size of the return buffer is specified in bytes instead of characters and the source/destination strings are Unicode or multibyte strings.</td></tr>
<tr><td><b>Exception Criteria</b></td><td></td></tr>
<tr><td><b>Solutions</b></td><td>
<table>
<tr><th>Solution Applicability</th><th>Solution Description</th><th>Solution Efficacy</th></tr>
<tr><td>When copying a string.</td><td>As a rule, you must ensure that the return string buffer is at least large enough to hold the specified maximum number of</td><td>Effective, but still requires care in checking sizes.</td></tr>
</table>
</td></tr>
</table>

| | | | |
|---|---|---|---|
| | | characters, not bytes, plus the NULL character.<br>Follow these rules for safe use of strncpy()<br>1. Verify that src and dest are not NULL.<br>2. Null terminate the final character of DEST.<br>3. Use strncpy(dest, src, sizeof(dest)/ sizeof(dest[0])).<br>4. If the final character (i.e., sizeof(dest) - 1) of DEST is no longer null, then the buffer was overrun.<br>If using the "sizeof" operator to allocate the destination string buffer, you should use something similar to "sizeof(lpString2)/ sizeof(CHAR)" or "sizeof(lpString2)/ sizeof(WCHAR)", depending on the target string type. For buffers that are not statically allocated, use an equivalent "sizeof" operator or constant that matches the declaration. | |

| | |
|---|---|
| | On Windows platforms, consider using StringCbCopyN (for byte counts) or StringCchCopyN (for character counts) from the strsafe.h library as safer replacements for strncpy(). These routines deal with NULL parameters better and ensure that the buffer is always null terminated. If you need to check the size of the input string to ensure that your destination buffer is large enough, you should use StringCbLength or StringCchLength to ensure that the buffer is the correct size.<br>On some UNIX platforms (FreeBSD), consider using strlcpy(), which also deals better with NULL characters. You still need to ensure that buffer size is correct. |
| **Signature Details** | Presence of the strncpy function. |
| **Examples of Incorrect Code** | `char str1[15];`<br>`char str2[20];` |

| | |
|---|---|
| | `strncpy(str1,str2,20);`<br>`/* The above will cause a buffer overflow on str1 as it can only hold 15 characters. Note that if str2 is null terminated and has 15 or fewer characters, strncpy() will pad the result with nulls out to 20 characters. */` |
| **Examples of Corrected Code** | `char str1[15];`<br>`char str2[20];`<br>`strncpy(str1,str2,sizeof(str1)/sizeof(str1[0]));`<br>`str1[sizeof(str1)-1] = '\0'; /* ensure null terminated */`<br>`/* The preceding is safe (though it will potentially truncate the string to be copied). If truncation is undesirable, should ensure that a sufficiently larger buffer is allocated. */` |
| **Source References** | • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/sec_winui.asp[2] |
| **Recommended Resources** | |
| **Discriminant Set** | **Operating System** — • Windows<br>**Language** — • C<br>• C++ |

## Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

[1] mailto:copyright@cigital.com